PROBLEM SOLUTION FIT- SMART SOLUTION FOR RAILWAYS

Date	**01-NOV-2022**
Team ID	**PNT2022TMID41891**
Project Name	**Smart Solution for Railways**

Abstract

The paper is concerned with the freight scheduling problem in the rail transport control systems. Its base sub problem of train
scheduling has been modeled as an assignment problem and solved using auction method. Results of comparison of auction
algorithm and Hungarian algorithm application for train scheduling problem showed that auction algorithm is significantly faster
in convergence. Towards the end, in this paper, was proposed the ways of further algorithm improvement.

Introduction

Freight scheduling problem is one of the most complex problems among those that arise in rail transportation
and therefore is in high demand. Freight scheduling problem includes three sub problems: train scheduling problem
(deciding on the most convenient timing for train departure along its route), locomotive assignment problem and
locomotive team assignment problem, i.e assigning locomotives to trains and teams to locomotives in the most effiffifficient
manner. Although these problems need to be solved all together, train scheduling problem is the base problem, on
solution of which restrictions that have to do with locomotives and teams are later on applied to.
In this article one of the problems currently researched for Russian Railways project will be discussed in detail. The
base problem of train scheduling will be modeled as an assignment problem and solved using auction method

Problem description

The main restriction in train scheduling problem has to do with the fact that arbitrary times of arrival and departure
of a train may not be assigned. According to Russian Railways technology trains may only be assigned to specifific
predefifined time slots that are considered input data to the problem. Slot information is usually provided for the next 24
hours. Therefore, train scheduling problem can be solved as an assignment problem, i.e. finding optimum assignment
of trains to slots.
Input to the scheduling problem includes the following:

• Initial locations
of trains
•
Information about trains
at stations that are ready to be departed
•
Information about trains that are not yet ready for departure with estimate of time needy to become ready
•
Time slots available for train assignment
For each train its current location and route until destination is given. The problem is to fifind an appropriate slot for
the train with given characteristics as part of optimal train assignment of all other trains to some existing slots. Note
that not necessarily one train would occupy the full length of a slot – partial occupation is allowed. Two or three trains
may occupy one slot (at different parts of slot route) or the contrary – one train may use more than one slot along its
route (different parts of train route).

Assignment problem description

Assignment problem is classical mathematical problem that exemplifies combinatorial optimization problem. Gen
eral case of such problem is described as the following:
•
There exist some number of tasks (
N) and a number of employees (
M
) that execute tasks. Each employee
performs
a portion (*P*) of all jobs (0
≤ *P* ≤ *N*), given that effifiifficienc
y of each employee when performing
assigned tasks is difffferent – some tasks he is more qualifified to do and some less. The problem is to distribute
the tasks among the employees in such a manner that the overall effifiifficiency is maximized.
For each < *task*, *employee* > pair there is a utility function *Ui j* which quantifies the effifiifficiency of assigning this
particular employee to this particular task. Thus optimization problem can be set up as maximizing overall effifiifficiency
Ui j of all assignments given that one employee is assigned to not more than task.
Lets consider how to formulate the problem of assigning trains to slots as a classical assignment problem:
• Trains would be considered as employees that need to be assigned to slots then the main requirement of this
assignment problem would be to make all employees busy with a task (all trains should be assigned to a slot).

• A task is equivalent to a slot then the condition that ”every employee can execute a number of tasks” is equivalent to
each train having a choice of what slot it can be assigned to. Moreover, some < *train, slot* > pairs are
impossible, for example, when route of a slot is totally diffffferent from train route and they have no common
partial route within their routes.
Each train route can be assigned to diffffferent slots throughout its route therefore the problem is fact a sequential
solution of assignment sub problems if routes of slots and trains are fragmentized.
• Each
assignment sub problem is an assignment problem for trains departing at a given station.
•
For each station a train set that needs to be assigned to slots should be determined as part of input to the sub
problem. At the fifirst iteration this set is comprised of all trains available for departure at a given station starting
from the beginning of the planning period and trains previously partially assigned and soon arriving at the given
station for the further route assignment.

• After
we solve one assignment sub problem, we have determined the time when each train would arrive at
the next station according to the trains route. Therefore at the next iteration all trains assigned at the previous
iteration would turn into ”trains previously partially assigned and soon arriving at a given station” at the next
station of train routes.
• Thus,
before starting a new iteration we calculate the number of trains that need to be assigned at each station
and a given station would be chosen as the one with the maximum number of trains to be assigned. After
iterating all stations, we’ll eventually assign each train route fragment to a slot.

4. Utility function calculation

The main diffiffifficulty in such problem formulation is determining utility function of each assignment pair < *train, slot* >.
The key issue is determining a rule to compute utility for each < *train, slot* > pair. We suggest proceeding with the
key issue of computing utility function in the following manner. First of all, we outline the main criteria that determine
the overall effiffifficiency of a < *train, slot* > assignment. Such criteria identifified are the following:
1. Train waiting time. Obviously, downtime of any train serves little purpose and the overall assignment should
minimize the overall downtime of trains. Here we have spotted a few nuances in the technology of Russian
Railways:

(a) This criterion should be taken into account but with a lower priority at stations where a train is formed rather
than at stations that a train is stopping along its route.
(b) For stations where locomotive team or locomotives are to be changed the time for such change should be
taken into account.
2. Regularity of hourly train departures has to be sustained, that is:
(a) The quantity of trains scheduled for departure at each hour should be close to the predetermined mean value
of trains to be departed at each hour from this station.
(b) Some available capacities should be left unassigned for emergency departures. For instance, if at a given
hour h there are 5 time slots available and there are only 2 available at hour $h + 1$, then the assigning time
slots at h is preferable.
3. Percent of compliance (number of matched fragments) between slot route and train route. Trains should better
be assigned to slots with a higher percentage of compliance between slot and train routes.
4. Continuing with previous train assignment. If a train is assigned to a slot at previous route fragment, then its
better to assign it to the same slot at the next fragment.
Each of the listed above criteria is put in correspondence with a normalized value uk, then the utility function value
for each < *train, slot* > pair is calculated as $Ui\ j = \ k\ ckuk$, where ck weight of k-th criterion. Weights can either be
chosen experimentally or be assigned a specifific value for each fragment.
Note that there is sense in calculating utilities only for such slots to which a given train can at least in theory be
assigned. For the remaining ”impossible” slots utilities are left unassigned. A slot can be considered ”impossible” for
a given train if train route and slot route have none fragments matching, or slots designed for a category of trains into
which the train does not fall. Set of such criteria can be assigned individually for each fragment.

5. Solving the assignment problem using auctions

An assignment problem is special case of transportation problem and transportation problem is in turn a special case
of linear programming problem. Such problems are solved with the classical simplex method, but using a specialized
method would yield faster results.
The most common way of solving an assignment problem is the ”Hungarian algorithm” which is a combinatorial
optimization algorithm. It was proposed in 1955 and its complexity is $O(n4)$, where n is the quantity of jobs and
employees (number of jobs has to be equal to the number of employees) 6. The algorithm can be modifified in a way
to reach $O(n3)$ complexity. The Hungarian method has been implemented for the problem of assigning slots to trains
but its speed proved to be unacceptable. Therefore another algorithm was discovered and the algorithm is solving the

assignment problem using auctions.
Solving an assignment problem using auctions was proposed by D. Bertsekas 2 in 1989 and later in 1992 it has been
modifified by Bertsekas and Castanon3. Its initial purpose was for solving skewed assignment problems where the
number of jobs is not equal to the number of employees. The main concepts of the auction algorithm are listed as
follows:
1. For each job (in our case slot) a concept of a ”price” *pj* is introduced. It shows the added ”price” that an employee
(in our case train) would have to pay in order to be assigned to the job (slot). All initial ”prices” are set to zero.
2. At the fifirst step a train is arbitrarily chosen and is assigned to a slot with the highest utility. Then the ”price”
of the slot with the highest utility is set to the difference between its utility and the utility of the ”second best
option”.
3. Lets say, for some other train the ”best” option is also the ”best” option slot from the previous train, then if this
lot is reassigned to this new train, the train previously assigned to it would have to be reassigned and the overall
utility would decrease by the ”price” of the best option, i.e the difference of the ”best” and ”second to best”
options for the first train. Therefore, for the overall utility not to decrease, it is necessary from the second train
its ”price” of the best option is higher than the cost of reassigning the first train.
4. So fifiguratively speaking the trains are now ”negotiating” their slot options but slot prices are ever changing.
Thus, the steps of the auction algorithm are:
1. For an unassigned train *i* the best option slot *j* is chosen: $ji = arg \max j \in A(i)\{Ui\ j - pj\}$, where *pj* is the price of
slot *j*, *A*(*i*) is the set of slots, possible for assignment to train *i*.
2. ”Second to best” option of a train to be assigned to slot *j* is calculated *ji*: $wi = \max j \in A(i), j \neq ji\{Ui\ j - pj\}$. If no other
3. After that new price for slot slot exists except for the assignment performed then *wi* is set to $-\infty$.
j is calculated: $pj = \max\{\lambda$
; *U*
$(i\ ji) - wi + \}$, where
λ
is a constant threshold value.
Assigning a price lower than the threshold is forbidden.

is an infifinitesimal of
1
N_N order.

4. If $\lambda \leq$
$u(i\ ji) - wi +$, then train
i is assigned to slot *j*, previous assignment to slot *j* is rolled back (train *i* , that

used
to be assigned to slot
j is deleted from the set of assigned trains).
5. Steps 1-4 are repeated until all trains are assigned to slots. If the number of slots is less than the number of trains
then at some step the algorithm would converge to its optimum assignment and the set of assigned trains would
stop enhancing. In this case the algorithm has to be stopped.
Mathematical proof of convergence of auction algorithm, as well as a guideline for choosing values for λ and are
given in3.

6. Results and ways of algorithm improvement

For the sake of comparison of Hungarian algorithm and auction algorithm both algorithms were implemented using
AgentSpeak. Our current implementation of auction algorithm is multi-agent. Main objects of the system are: slots
agents, stations agents as well as planner agent serving for central synchronization of all system elements and forming
fifinal decision.Station agents store information about assigned trains and regulate the order of solving assignment sub
problems. Planning complexity is occasioned by great amount of participators with conflflicting interests, variety of
criteria for decision making, uncertainty, dynamics and other factors. Adaptive real-time plans creating and realization
is significant advantage of multi-agent approach in dynamic resource management systems planning and optimiza
tion. Testing showed that for problems of size ~ 200
$* 200$ auction algorithm is signifificantly faster in convergence
than Hungarian algorithm: 20 seconds versus $3 -$
4 minutes (given time parameters should not be used as a refer
ence for comparison with performance of algorithms implements using other programming languages or frameworks.